2019

Combining evolutionary computation with the variable neighbourhood search in creating an artificial music composer

Reza Zamani
University of Wollongong, reza@uow.edu.au

Combining evolutionary computation with the variable neighbourhood search in creating an artificial music composer

Abstract

This paper presents a procedure which composes music pieces through handling four layers in music, namely pitches, rhythms, dynamics, and timber. As an innovative feature, the procedure uses the combination of a genetic algorithm with a synergetic variable neighbourhood search. Uniform and one-point crossover operators as well as two mutation operators conduct the search in the employed genetic algorithm. The key point with these four operators is that the uniform crossover operator and the first mutation operator are indiscriminate, in the sense of using no knowledge of music theory, whereas the employed one-point crossover operator and the second mutation operator are musically informed. Music theory is used for finding the suitability of its generated pieces. The method starts with generating an initial sequence of pitches with a musically informed module and then calculates the suitability of the pitch sequence through the embedded rules. The employed genetic algorithm applies the variable neighbourhood search method to its generated offspring genomes for increasing their quality. Pieces can be composed in major, minor, and harmonic minor scales based on the user's request. As well as composing the main notes, the procedure generates up to three chord notes associated with each main note and plays the result in a novel multithreading environment through running four threads concurrently.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Zamani, R. (2019). Combining evolutionary computation with the variable neighbourhood search in creating an artificial music composer. Connection Science, 31 (3), 267-293.

Combining Evolutionary Computation with the Variable Neighbourhood Search in Creating an Artificial Music Composer

Reza Zamani, Ph.D. (OR)

School of Computing and Information Technology, Wollongong University, Wollongong, NSW 2522, Australia

E-mail: reza@uow.edu

Abstract: This paper presents a procedure which composes music pieces through handling four layers in music, namely pitches, rhythms, dynamics, and timber. As an innovative feature, the procedure uses the combination of a genetic algorithm with a synergetic variable neighbourhood search. Uniform and one-point crossover operators as well as two mutation operators conduct the search in the employed genetic algorithm. The key point with these four operators is that the uniform crossover operator and the first mutation operator are indiscriminate, in the sense of using no knowledge of music theory, whereas the employed one-point crossover operator and the second mutation operator are musically informed. Music theory is used for finding the suitability of its generated pieces. The method starts with generating an initial sequence of pitches with a musically informed module and then calculates the suitability of the pitch sequence through the embedded rules. The employed genetic algorithm applies the variable neighbourhood search method to its generated offspring genomes for increasing their quality. Pieces can be composed in major, minor, and minor harmonic scales based on the user's request. As well as composing the main notes, the procedure generates up to three chord notes associated with each main note and plays the result in a novel multithreading environment through running four threads concurrently.

Keywords: Genetic Algorithms, Multithreading, Artificial Intelligence, Music Theory, Evolutionary Computation, Artificial Music Composition, Local Search, Variable Neighbourhood Search, Local Optimality

Introduction

Algorithmic composition is an array of techniques towards using algorithms for creating music. A point which eases such algorithmic composition is that mathematics plays a key role in music theory. Indeed, the link between mathematics and music theory dates back to the time Pythagoras (570 – 495 BC) developed a conjecture of consonants based on ratios of minute integers, and during these twenty-five centuries mathematics and music have been closely entangled.

In effect, among the territories enhanced by computational intelligence, art, because of its extreme sophistication, is of paramount importance. The issue is not simply that with conquering this territory, computers can blur the boundary dividing fantasy from reality; the deeper issue is that in this way computational intelligence penetrates the area of imagination, which is the source of creativity and hence promotes the understanding of human creativity.

Recently, the focus on computer applications to art has significantly increased. For instance, aimed at understanding human creativity and capturing it through algorithmic approaches, (de Vega et al., 2014) have presented an evolutionary procedure for emulating human creativity in art. This procedure, which emulates creativity, can allow a team of artists collaborate and affect its performance. Two algorithms based on swarm intelligence for dealing with computational creativity have been introduced in (al-Rifaie, Fol Leymarie, Latham, & Bishop, 2017).

(Loaiza, 2016) has proposed a way to understand the confluence of the enactive approach to cognition and musicology in a wider sense, and in (Boden, 2016) skills and appreciation of computer art has been extensively described. However, a wide range of algorithmic matters must be dealt with for relatively full appreciation of computers in

potential sectors of art.

Consistent with the universal acceptance of compositional rules in music generation, based on their extensive studies, (X. F. Liu, Chi, & Small, 2010) have concluded that "good" music displays some universal features independent of cultural issues.

Computational creativity is the dream of Artificial Intelligence researchers, and can be used in a variety of ways in producing artistic objects, ranging in areas from industrial design, through music, to architecture. Among them artificial composing is of significant prominence. The reason is that not only research on automatic composition can add a considerable variety to the current interesting music pieces but it can deepen our understanding from music composition as well.

As sophisticated formulas in physics are the valuable work of physicists, and algorithms can signify the precious work of artificial intelligent scientists, symphonies can be considered as valued artefacts of musicians. Ingenious ideas of artistic artefacts, like the creation of symphonies, when captured in algorithms, represent the golden braid of art and artificial intelligence,

There is always a time-lag between the time a technology is introduced, and the time it is fully utilized; AI as an algorithmic technology, which is aimed at mimicking human creativity, is not an exception. What distinguish AI from other technologies are, however, two related facts: first it programmability promotes the performance of a large variety of creative tasks, and second, creativity has no perceived limit.

That is why computational intelligence is expected not only to duplicate human creativity but to expand such creativity beyond the capability of human mind. In this regard, the composition of music is an example. The development of an effective

interactive composer's assistant software applications is not, however, comparable to that of word processing software applications.

The reason is that the major operations of word processing like deletion, copy, paste, save, open, and print are only a small part of such composing applications, as these applications are involved with many other layers. Graphical, audio, and composition interfaces are examples of such layers, with composition interfaces focusing on both computational analytic of the passage and subtleties involved with the innovative task of composition.

In fact, such composing applications need to provide solutions to different compositional problems by providing musical ideas and broaden the collection of innovative artefacts of the composer. Computer Aided Design (CAD) applications, in comparison to word processing applications, are more similar to Computer Aided Composition (CAC) and it seems that after the great success of CAD in the previous decades, CAC can be considered as the next candidate for gaining such success, necessitating further research on all related areas from editing, through structuring, to composing.

With respect to such necessity, this paper discusses the development of an effective artificial music composer. The presented procedure enforces coherence and unity in its generated pieces through combining three concepts in innovative fashions. The first concept is to select a key note and emphasizing on such a note which moves dynamically. The second concept is the use of succession of notes in interconnected circles, and the third concept is the use of symmetry.

What makes these transformations versatile is that they can be applied successively. For instance, the same as in geometry, rotation can be achieved by the combination of

horizontal and vertical mirroring, or the horizontal mirroring can be combined with the vertical and horizontal shifting. All of these transformations keep the original motif isometric, in the sense that the distances between the succeeding notes of the new motif is the same as those distances in original motif. As an eyelet to the world of composition, the procedure is aimed at developing comprehensive music composition. Because of using the concept of symmetry, and sticking to the phenotype of notes and durations, the procedure has been called SYMPHONY (Symmetric Phonotype Eyelet).

By entangling intervallic structures in the rhythmic ones, the SYMPHONY integrates time with pitches in meaningful patterns and avoids the pitfall of mechanical combination of pitches and time. In other words, every chunk of intervals generated with the same algebraic relation is treated as a whole and receives its whole rhythm, which is represented by an array of time.

Relying on consonance, dissonance, resolution and chord progression concepts, the SYMPHONY composes a piece as the aggregation of synergetic parts in depth, rather than separated entities in surface, relying on the fact that creative process of compositions is concerned with investigating effective musical notations and integrating them into highly deeply-rooted coherent pieces.

The SYMPHONY is a hybrid of several effective approaches in artificial composing of music. This approaches range from employing optimization techniques, through employing music theory, to the use of MIDI in producing several simultaneous voices with various musical devices. It has been designed based on the fact that computational intelligence not only can generate music automatically but can assist composers for generating high quality music, as well. The employed genetic algorithm uses a variable neighbourhood search for increasing the quality of offspring genomes created. This

search uses two neighbourhood schemes in a cycle, in the sense that after making a pitch sequence with the first neighbourhood scheme, the result will become local optimised through the second scheme, continuing this until neither scheme can make any improvement in the pitch sequence. Figure 1 shows schematic representation of the SYMPHONY and the machinery it employs for generating and playing music.

Figure 1 is inserted here.

The SYMPHONY relies on music theory, and can compose a piece without the interference of a human evaluator, circumventing the traditional shortcoming of decreasing sensitivity which occurs at prolonged listening of evaluators to the generated pieces. It composes by arranging a set of notes and then assigning duration to each note so that the integrated result is pleasing. Factors contributing to such a proper arrangement ranges from satisfying created expectation, through constructing intervallic symmetry, to providing intricate variation for mildly violating the created expectation. In fact, it is the proper combination of these factors which makes the SYMPHONY effective and versatile.

The rest of the paper is as follows. Section 2 discusses the related work in seven different subsections, namely (i) the relationship between mathematics and music, (ii) music and NP-completeness, (iii) pattern extraction in music, (iv) quantifying listeners' expectation with using the concept of entropy, (v) music composition with Markov chains, (vi) music composition with variable neighbourhood search, and (vii) music composition with genetic algorithms. Section 3 describes the SYMPHONY by presenting its constructing components consisting of (i) the music-rules-handling module, (ii) the pitch, rhythm as well as chord generation modules, and (iii) the modules for timber and dynamics modification. Section 4 describes the examples generated by

the SYMPHONY, and novel points regarding their generation. Section 5 presents concluding remarks and future research directions.

2. Related Work

Computational music theory presents many interesting combinatorial, geometric, and algorithmic problems useful for both the analysis and automatic generation of music. Several of these techniques have been surveyed in (G. T. Toussaint, 2003), a number of combinatorial problems that either have been suggested by musical topics or have arisen in music theory have been surveyed in (Read, 1997).

(Munoz, Cadenas, Ong, & Acampora, 2012) have considered music composition as one of the endeavours which can greatly benefit from computational intelligence, proposing an intelligent scheme which can compose music based on an optimization problem and solving it with an evolutionary technique. This technique works based on an adaptive multi-agent memetic approach, with an optimization process arranging chords with the main notes of the pieces.

Manipulating the surface structure of music ignores the fact that it is only the music theory underlying a particular knowledge representation scheme which can lead to enhancing the descriptive power of a representation (Hirata and Aoyagi, 2003). Algorithmic manipulation of the deep structure of music, on the contrary, is linked with emulating human creativity. This link can be further emphasized by noticing that it is ingenious thinking which leads to brilliant and incredible music composition. This affects a variety of algorithmic music endeavours ranging from the recognition of musical work through music composition to sound synthesis and sampling.

Music composition because of its interconnection with intelligence and the creativity

of the composer can greatly rely on computational intelligence for automatic composition. Indeed, the idea of automatic composition dates back to the time Mozart who introduced his famous Dice Game **(**Musikalisches Würfelspiel**)**. In this game, small musical fragments were stochastically combined to create a larger fragment (Herremans and Sörensen, 2012).

The related work has been divided into seven subsections, namely (i) the relationship between mathematics and music, (ii) music and NP-completeness, (iii) pattern extraction in music, (iv) quantifying listeners' expectation with using the concept of entropy, (v) music composition with Markov chains, and (vi) music composition with variable neighbourhood search, and (vii) music generation with genetic algorithms.

2.1 The Relationship Between Mathematics and Music

The benefits of applying computational and mathematical approaches to the field of music as well as the pitfalls in such applications have been discussed in (Mazzola, 2012), emphasizing on the challenges for strengthening the connections among musicological, mathematical, and computational approaches. The reason such approaches are successful when applied to music is that musical, computational, and mathematical concepts have a number of construction principles in common, with mathematical and computational models facilitating effective extension and generalization of music pieces.

(Papadopoulos, 2014) has a quick overview on the relationship between mathematics and music has been presented, emphasizing on the rule of group theory in music through using several examples selected from the composition pieces of Olivier Messiaen (1908-1992), who was one the influential composers in twentieth century.

Through presenting an extensive list of references, its author has named the other fields of mathematics, besides group theory, which are involved with music, as probability, combinatorics, geometry, and graph theory. Considering consonance as a puzzling question regarding how playing some different sounds together seems pleasing, he has named Aristotle, Euclid, Galileo, Kepler, and Euler among those who have written about this topic.

Reviewing recent results on the computational aspects of rhythms and melody, (G. Toussaint, 2010) proposes connections to the areas like computer science and mathematics. At the heart of algorithms designed for measuring the similarity of rhythms, the author has distinguished the swap distance, the directed swap distance, and the many-to-many matching distance.

Whereas the swap distance method, which is used for the rhythms which have the same number of onsets in their binary representation, measures how the sum of mismatches between every two corresponding onsets is, the directed swap distance captures the same essence for the cases where the number of onsets are different. The many-to-many matching distance removes a shortcoming of the directed swap distance in measuring the sum of mismatches between every two corresponding onsets for the cases in which the number of onsets are different.

With respect to tiling rhythmic cannons as a mathematical problem, (Andreatta, 2011) has traced back its history of algebraic formalization. The dynamism between a given musical problem and its mathematical statement has been considered, showing that as intervallic structures, rhythmic structures can also be transformed. Also the same with intervals, rhythms which cannot be retrograded or transposed contain small retrogressions or transpositions, respectively. In this regards, rhythmic cannon has been

considered as a repetition with temporal translation of a simple or transformed rhythmic structure.

2.2 Music and NP-Completeness

(Gwee, 2013) has brought computational science into picture through considering composition and music theory within the domain of NP-complete computational models. As an example of these NP-complete problems in music, the author has mentioned the production of counterpoint, as a polyphonic music composition, based on a given primary melody referred to as 'cantus firmus'. In other words, the author has mathematically proved that the production of counterpoint is an NP-hard problem. Heuristics, genetic algorithms, and artificial neural networks have been suggested to do polyphonic music composition.

2.3 Pattern Extraction in Music

The discovery of recurrent melodic patterns in music analysis, in general, and music composition, in particular, plays a key role, and sequential data mining can be used for pattern extraction purposes. Represented by a prototype, a melodic pattern refers to an abstract entity, like regular expressions, to capture the concept of semi-identical structures (Rolland, 1999).

Considering the quality of music as evoking the sense of movement in the listener and relating this feature to meter and rhythm, (Eck, 2001) has explored the details of a rule-based model predicting downbeat location as well as pattern complexity in rhythms.

Sequences of notes which are defined more than once in a piece are called repeating patterns. However, sometimes a sequence after slight changes is repeated. Considering the discovery of non-trivial repeating patterns as a basic characteristic of music analysis

and content-based music retrieval, (Hsu, Liu, & Chen, 2001) have developed two data structures, namely correlative matrix and string-join-operation, each used in one of their two approaches.

Computational music analysis is involved with automatic detection of recurring patterns in music. With highlighting the fact that knowledge representation techniques in music should have the capability of expressing common abstract patterns, (Conklin, 2002) have described a new technique which can discover patterns in both vertical and horizontal axes of polyphonic music.

Emphasizing on the existence of large databases of music data and the importance of approximate pattern matching in music information retrieval and analysis, (Clifford and Iliopoulos, 2004) have presented an overview on the advances made in the corresponding area, focusing on innovative measures of approximation.

With two examples taken from African traditional music, (Chemillier, 2004) has illustrated cyclic shift and its role in enhancing musical structure, emphasizing how two cyclic shift solutions can be considered the same. Since two periodic sequences cannot be distinguished when the listener misses a particular number of notes in one of the two sequences, the idea defining an equivalence relation on periodic sequences has been captured by the concept of conjugacy relation.

Human can detect various similarities in music both in rhythmic and intervallic structures. With respect to comparing the similarities between any two melodies, (Aloupis et al., 2006) have parented efficient algorithms which compute minimum area between two polygonal chains, representing the two melodies in time and pitch dimensions.

(Demaine et al., 2009) have demonstrated the relationship between distance geometry in music and the classic Euclidean algorithm, and then have defined a family of rhythms by using this algorithm. The authors have considered both time and pitch dimensions of scales with cyclic nature, in the sense that pitch dimension cycles in every octave and time dimension cycles in every measure. To relate rhythm with this cyclic nature of time dimension, they have used the term rhythm to mean timeline.

Timelines, which are also called claves, are often played with instruments producing unsustained notes and accentuation is used to determine the start of each clave. In most interesting rhythms, the number of onsets, k, and time-span, n, are relatively prime, in the sense that, their greatest common devisor is 1. For instance, this is true for (3,3,4,2,4) in which the number of onsets, 5, and time-span, 16, which is equal to 3+3+4+2+4, are relatively prime.

For melody music objects, (Karydis, Nanopoulos, & Manolopoulos, 2007) have introduced an effective procedure for discovering patterns with maximum length. Their proposed algorithm avoids the examination of intermediate patterns, whose number is large, and only concentrates on finding the maximum-length patterns. These patterns can be later employed by expert systems for composing music.

In (Halkiopoulos and Boutsinas, 2012), for developing a music improviser, machine learning has played the role of extracting regularities and patterns from data. For instance, the association rule mining can state that A#, C → B, which means after the two successive notes of A# and C, it is the note B which appears. Their proposed procedure divides the input melody into consecutive fragments which have the same direction. Hence, the number of these fragments depends on how frequently the notes have changed their direction. Edit-distance measure has been used for pattern matching,

and based on the distances calculated, the patterns can be clustered. The most similar pattern to each of the fragments is identified and replaces the corresponding fragment.

2.4 Quantifying Listeners' Expectation with Using the Concept of Entropy

Emphasizing that the expectation of hearing a particular note depends on the listener's model of the genre music and is not intrinsic to the musical property, (Witten, Manzara, & Conklin, 1994) have used the concept of entropy to quantify such expectation. In effect, the prediction of upcoming events depends on what the listener has already listened, and entropy, which is expressed in terms of bits per event, has been considered as a proper criterion to measure predictability.

For instance, for an event which has the entropy of 1 bit, a predictability of 50 percent and for an event which has the entropy of 2 bits the predictably of 25 percent is envisaged, with an event which has the entropy of zero being hundred percent predictable. In this way, entropy, which is a real number, measures how predictable the notes in a piece are, and can compare two pieces.

In comparing human models of music prediction with machine-based predictive models, with respect to Bach chorale melodies, they have measured that for Chorale 161, the average entropy of pitches is 1.97 bits per event using human model, and 2.09 for their computational model. Their extensive comparisons indicate that currently people predict pitches better than machines.

However, there is no reason that situation remains the same in future. Indeed, algorithmic and technological improvements can assist machines to enhance their capabilities for this purpose, and as the authors have stated, it seems that, eventually, machines, at chorale guessing game, will outperform humans.

804 pieces of classical music written by twenty-nine different composers, have been considered in (Volchenkov and Dawin, 2012), and encoded into transition matrixes. Then their Markov chains have been extensively studied. Based on these experiments, it has been shown that regardless of composers, the number of pitches used in each composition grows approximately logarithmically with the length of composition. Shannon entropy has been used as a criterion for measuring the uncertainty in the occurrence of a pitch in a stochastic process.

For all these 804 pieces, the magnitude of entropy has fluctuated in a range between 0.7 and 1.1 bit per note. It should be noted that when the entropy is 1, among 2 guesses about a note, one of them is correct. An important point with transition matrixes created is that they all are of the size 12*12.

The reason is that in line with many other researches, the authors have agreed upon the fact that in tonal music, notes spaced over several octaves are perceived nearly the same as if they are played in the same octave. The authors have concluded that the frequency analysis of note occurrences cannot, as a single factor, resolve the tonality of a composition.

2.5 Music Composition with Markov Chains

Considering the fact that Markov chains can in general produce variations on a single theme of music and the range of these variations can be controlled by changing the cells of the corresponding matrix, (Shan and Chiu, 2010) has integrated Markov chains with genetic algorithms to create a procedure for the music composition purpose. Whereas Markov chain is used to select the next rhythm, pitch, and chord, the genetic algorithm searches for the set of Markov chains which can produce more pleasing music. Their genetic algorithm requires a human listener to operate as its fitness function evaluator.

Classifying machine-generated music composition into two approaches of taking rules from a piece or taking them explicitly, (Shan and Chiu, 2010) have presented a top down approach for discovering particular rules of music composition from an inputted piece and then they have used those rules in creating similar pieces.

In effect, by using data mining techniques, they have presented a top-down approach in discovering the rules of musical composition in a given piece for the purpose of applying the discovered rules in creating similar pieces. As a top down approach, this method pays special attention to the beginning and ending of a piece. Hidden Markov Model has been considered as the most common approach for automatic music composition.

Considering the fact that Markov hypothesis determines the future state of a sequence based on only the last state, a memristor network, as a step beyond Markov's chains, has been proposed in (Gale, Matthews, Costello, & Adamatzky, 2013). A memristor, as a fundamental circuit element, changes its resistance proportional to voltage and as a function of the charges which has passed from it. Upon presenting such a system for music composition, the von Neumann's hardware limitations for simulating such a composing memristor network has been discussed in detail. The authors have also proposed a hardware solution based on considering the physical properties of the presented memristor network for composition.

In the direction of analysing the tonal behaviour of a music piece, (Tardón, Barbancho, Barbancho, & Roig, 2014) have presented a method which uses probability model with a well-known chroma descriptor. This probability model, which is used for key analysis, is aimed at analysing the presence of each of the pitches in various tonal keys for a given piece.

2.6 Music Composition with Variable Neighbourhood Search

In (Todd and Werner, 1999), efforts towards generating algorithmic music composition have been reviewed and the way these algorithms discover patterns, follow prescribed rules, and evolve towards being matched with some aesthetic criteria have been discussed. The authors have classified algorithmic composition techniques, in the order of their introduction into the three categories of (i) formal rules, (ii) example learning, and (iii) modification along with evolutionary descent processes.

An interesting point with respect to the second class, which is “example learning”, is that as a human composer when exposed to low and high quality music learns to compose more effectively, a learning procedure when exposed to different pieces with various qualities, can discover patterns needed for high quality composition.

Search methods for music composition are multifaceted. Relying on expert systems, symbolic numeric machine learning, and composition theory, (Bel, 1998) has presented a theoretical framework in modelling descriptions of the compositional process. The issue of how polyphonic structures can be shown with linear text format has been surveyed and its complexity has been emphasized.

Emphasizing that composing music can partially be considered as a combinatorial optimization problem, a variable neighbourhood search has been developed in (Herremans and Sörensen, 2012), which generates the first species counterpoint for any input melody. Assuming there is a single melody, called *cantus firmus*, the counterpoint starts from such a fixed song and adds a second melody to it. In their procedure, melodic rules govern the construction of such a second melody. This optimization technique starts with a stochastically generated melody and, by changing one or two notes at time, enhances it towards becoming a counterpoint of the input

melody. The employed variable neighbourhood search has been compared with a genetic algorithm and has been shown to be more efficient.

As well as the variable neighbourhood search, a backtracking-based depth-first search which can build counterpoints with up to 3 voices (Alarcón, 2013) has also been used to tackle the same problem. Compared to the procedure used in (Herremans and Sörensen, 2012), which uses the variable neighbourhood search, this procedure uses more rules and hence is more restrictive. Because of this restrictiveness, the procedure sometimes fails to produce a valid counterpoint and has to abandon its search, and start from scratch. The length of prices generated is between 7 and 17 notes.

In (Jo, Kim, Kang, & Lee, 2007) a chord-based music composition procedure has been developed, which is entirely different from a melody-based composition, and composes music through the use of various chords. Selecting the top ten music within a period of 5 years, the authors have identified a set of possible chord progression rules and embedded them in a probabilistic framework. Guided by this framework, the procedure proceeds from one chord to another and composes a piece.

Without any requirement for subjective feedback of human evaluators, the (Munoz, et al., 2012) have developed an artificial composer, which uses music theory, and music charts for its evaluation purposes. In particular, weighted rules from music theory which are obtained by downloading music charts, guide the composition performed by the procedure. The employed representation divides each octave into twelve standard notes and assigns numbers from 0 to 11 to these notes, with 0 being assigned to C, and 11 being assigned to B. 42 Evaluation rules, each with different weight, are used to measure the fitness of sequences produced by the procedure. The weights of rules, which can also be negative, have been calculated through downloading information

from music charts.

For this purpose, 33 songs have been selected and depending on their ranks, which have been determined based on the times they have been downloaded from a particular web site and the number of times each song has used the considered rules, the weights of the rules have been calculated. For calculating weights, a system of linear equations has been constructed in which the times each rule has been used for each song are coefficient multipliers, with downloading time for each song being the right hand side of each equation.

2.7 Music Composition with Genetic Algorithms

The first published state-of-the-art GA technique applied to music composition has been reported in (Horner and Goldberg, 1991). In this work, the implementation of a GA-optimizable operation set for music composition has been grounded on a linkage building block and the results have been discussed, emphasizing on the possible future work in music composition through GAs.

In general, problem representation plays a key role in GAs, and a GA cannot be effective when using ineffectual coding-decoding mechanism. The problem of music representation has been considered in (Smaill, Wiggins, & Harris, 1993) and a generalized hierarchical structure has been proposed, suggesting that significant grouping of musical events is considered as an advantageous point in such representation.

(Moroni, Manzolli, Von Zuben, & Gudwin, 2000) have presented a genetic algorithm called *Vox Populi*, which means “voice of people”, that composes music in real time. The pool of this genetic algorithm includes a population of chords which undergo

genetic operators. Physical factors related to music comprise the corresponding fitness evaluation function. In their approach, each chord consists of four notes, and hence sequences of four-notes are survivors of the process. Because of its real time nature, among the produced chords, the most fitted chord is played and the search continues for finding the next chord to be played. Based on the last chord played, survival condition is modified so that played chords are in accord with one another. As is seen, unlike in the SYMPHONY, in this work composition is performed through chords.

As with the above procedure, the procedure presented in (Moroni, et al., 2000) produces real-time chords using a genetic algorithm. The difference is that it approximates the sequence of notes to a note which shows the tonal centre of the sequence. Given a sequence of notes, the tonal centre is a note which is most consonant with all of notes of the sequence. With converting a chord to its tonic centre, harmonic and melodic fitness values govern the selection of the played notes. Numerical theory of consonance formed the employed fitness function.

(Dahlstedt and McBurney, 2006) has assessed the degree in which an agent paradigm can be applied to music, leading to better understanding of the process in which music is composed. In their approach, agents, as autonomous software entities which can co-operate with one another, exhibit kinds of intelligent behaviour in composing music. The authors have also developed a prototype software application which can support composers, improving techniques and tools assisting the generation of appealing music. They have stated that the project of developing infrastructure for their software prototype had been more difficult than they had anticipated and emphasized on the need for further development of such infrastructure. Immaturity of agent paradigm and hence the lack of proper design tools has been considered as main hindrance of developing

such tools.

Using a black-box optimization model employing evolutionary computation, (Chen, 2007) has presented an innovative framework for evolutionary music composition. In this framework, the user can compose customized music by being assisted through interaction with the system. Using two levels of hierarchy, the procedure is aimed at producing short length pieces. The evolutionary operators involve appending, inserting, merging, and splitting notes as well as doubling and shortening the time of a note.

Whereas the append operator concatenates two music blocks, the insert operator puts a block inside the other block. The merge operator combines two notes into a single note and the split operator decomposes a note into two different notes. The double and shorten operators change the time of every note of the block they are applied to by doubling and halving them, respectively. Unlike this procedure, the SYMPHONY does not need user's interaction for creating a composition.

In attempting to produce music artificially, (X. F. Liu, et al., 2010) have constructed networks in which nodes represent notes and edges represent co-occurring connections between the notes, analysing Chinese pop music. They have reported remarkable similarities among the networks created and conjectured that in the artificial composition of music, the preserving of the universal network properties is a crucial step. Controlled random walk, which begins from a node in the network and with biased random rules visits subsequent nodes, is the major mechanism the authors have used.

Classifying machine-generated music composition into two approaches of taking rules from a piece and taking them explicitly, (Shan and Chiu, 2010) have presented a top down approach for discovering particular rules of music composition from an inputted piece. They have used those discovered rules in creating similar pieces.

In (Diaz-Jerez, 2011), approaches have been discussed in which effective strategies in algorithm design have been used towards investigating musical structures serving as a framework for creative process of composition. The term Melomics, in which Mel perfix stands for melody and the omics suffix comes from genomics, has been used for technologies implementing simulated evolution of music compositions. In Melomics, natural selection shapes the musical structure of the scores toward generating more complex and sophisticated compositions.

(Munoz, et al., 2012) have considered music composition as one of the endeavours which can greatly benefit from computational intelligence. In this direction, they have proposed an adaptive multi-agent memetic approach which can compose music based on an optimization problem solved with an evolutionary computation technique.

Discussing computational intelligence and creativity and emphasizing on the roles of automatic accompaniment, (C.-H. Liu and Ting, 2012) have presented a genetic algorithm which generates polyphonic accompaniment. In this algorithm, several evaluation rules, all based on music theory, guide its customized fitness function. This fitness function which is constructed after analysing the rhythm and pitch structures of the corresponding melody, both reinforces the rhythm and harmonizes the produced music.

Using smart and musically informed operators, the genetic algorithm proposed in (Vargas, Fuster, & Castanón, 2014), uses musical theory in generating innovative melodies. Since the smart operators are musically meaningful, a majority of unfruitful parts in the corresponding search space are ignored.

3. SYMPHONY

In composing a piece, the artificial music composer presented in this paper, SYMPHONY, performs the following tasks (i) reading the input parameters, (ii) calling the genetic algorithm component and its variable neighbourhood search, (iii) selecting the best pitch sequences in the final pool of the genetic algorithm, (iv) randomly selecting a number of rhythms from the rhythm dataset for mixing or constructing new rhythms from scratch, (v) combining the best sequence found in the genetic algorithm with rhythms constructed, (vi) adding chords to the phrase generated, (vii) modifying the phrase in order to create three other different phrases, (viii) playing all four phrases, one after another, with their chords and showing the main notes on the screen while playing.

Since representation is the key issue of any genetic algorithm, first the representation scheme with the SYMPHONY is discussed. Depending on the user's choice, the SYMPHONY can compose pieces in the major, minor, and minor harmonic scales, using both step and semitone in measuring the distance of pitches. It is worth noting that when music is written tonally in a specific scale, a pitch can be measured both in terms of semitones and the number of steps in the corresponding scale. For instance, while G is 5 semitones away from D, i.e. D-D#-E-F-F#-G, it is only 3 steps away from D in scale C, i.e. D-E-F-G. In measuring the distance between notes, the SYMPHONY can use (i) semitones, as real transformation, and (ii) steps, as tonal transformation. It is worth noting that tonal transformation is the most common way to visualize music and whereas real transformation uses all 12 steps of chromatic scale, tonal transformation uses only 7 steps of diatonic scale.

Pitches, rhythms, dynamics, and timber are four dimensions which the SYMPHONY

handles in composing a piece, with dynamics showing the intensity of volume and timber showing the instruments by which the notes are played. The size of notes to be composed is an input variable and are determined based on the fact that many of famous pieces owe much of their fame to the use of short, well-balanced, and elegant motifs which intelligently are modified and repeated successively during the corresponding piece.

Like all other GAs, the proposed GA has (i) a representation scheme, (ii) a pool of individuals shown based on the employed representation scheme, (iii) evolutionary operators of crossover and mutation, (iv) a fitness function, and (v) a selection method. In general, musical knowledge can be represented either without encoding (direct) or with some encoding (indirect). An example of indirect representation is binary encoding, which through a decoder can be converted to direct representation. Among these two options the SYMPHONY uses the first alternative, direct representation.

The pool of individuals, in which every individual is an arrangement of pitches, can be constructed through random mechanisms, predesigned rules, or available melodies. In this regard, the employed genetic algorithm constructs the initial pool through a random mechanism. Evolutionary operators of crossover and mutation can be either musically informed or indiscriminate, the employed genetic algorithm uses two opposing uniform and one-point crossover operators as well as two opposing mutation operators. The term opposing has been used to emphasize that whereas the employed one-point crossover operator and one of the mutation operators are musically informed, the uniform crossover and the other mutation operator are indiscriminate. It is the combination of indiscriminate and musically informed operators which gives the necessary diversity to the pieces produced. Figure 2 shows the pseudocode of the SYMPHONY and that of

its genetic algorithm component.

Figure 2 is inserted here.

Towards imposing symmetry, the SYMPHONY enforces climax through chaining small fragments which have the same overall melodic direction. When leaving such a climax, other fragments, all with the opposite overall direction, are chained and move the notes in opposite discretion. The term overall has been used to indicate that not all notes need to follow the same direction but it is the summation of intervals which should be of the same sign. The employed music rules guarantee that pieces not conforming to having a climax to have a very small chance for being selected.

With respect to handling music rules as well as using intervallic, rhythmic, and chord structures, the following subsections describes the modules of the SYMPHONY which include (i) the music-rules-handling module, (ii) the pitch, rhythm and chord generation modules, and (iii) the timber, dynamics, and other modification modules.

3.1 The music-rules-handling module

Producing pleasing sounds through music without obeying musical rules is akin to constructing correct sentences in a language without observing its grammar. That is why the employed variable neighbourhood search uses a *music-rules-handling module* to construct its objective function.

In its music-rules-handling module, the SYMPHONY considers music multilayer, with viewpoints modelling its different layers and providing opportunity of looking at music from different perspectives (Conklin and Witten, 1995).

Pitch, as the basic concept in music, represents twelve equally distributed semitones and is referred to as note as well, with the ratio between wave frequency of two

consecutive note being equal to $\sqrt[12]{2}$. With setting A4 as 440 Hz, all devices assign the same frequencies to the same notes. This implies that when a note proceeds to one octave higher, by forwarding 12 semitones, its frequency is doubled.

The employed music rules, in general, and those used by the SYMPHONY, in particular, are mainly attributed to Fux, who is considered as the most prominent feature in music with having Hydn, Mozart, and Beethoven as his foremost disciples. His authority can be further emphasized by the fact that Haydn, who was in the same school whose director was Fux, largely taught himself counterpoint by reading Fux's work, and then recommended it to Beethoven who was younger than himself. Even Mozart had a copy of Fux's work and annotated it. Published in 1725, Fux's work has somehow revolutionized music theory and has laid a necessary foundation for artificial composition.

With using these rules, the SYMPHONY is aimed at holding the interest of listeners through smoothness, balance, proportion, and consonance. This is obtained through constructing an objective function which is enforced by its variable neighbourhood search component and discourages undesirable events. Table 1 presents the definition of terms needed to understand the events discouraged, and Table 2 describes the penalties associated with these discouraged events. Any other event can be discouraged by adding it to the list. As is seen, when an event is supposed to be encouraged, its negative event is discouraged. For instance, we can encourage to have at least two retrogrades with specific sizes by assigning penalties to non-confirming cases.

Tables 1 and 2 are inserted here.

3.2 The pitch, rhythm and chord generation modules

The pitch generation module, as the most critical module of the SYMPHONY, works based on the variable neighbourhood search and composes a series of musically-informed intervals of pitches, enforcing musical rules through minimizing the degree of dissonance, shown as penalties represented in Table 2.

The variable neighbourhood search is used in the employed genetic algorithm to increase the quality of the offspring genomes created. The pseudocode of this module has been presented in Figure 3. With using this module, the SYMPHONY calculates the suitability of a generated piece, and changes the piece through its embedded rules described in the music-rules handling module.

Figure 3 is inserted here.

The employed variable neighbourhood search uses two alternating neighbouring schemes for defining local optimality. For balancing exploration versus exploitation, the first neighbourhood scheme is basic, in the sense that it does not use music information whereas the second neighbourhood scheme is musically informed. Figure 4 shows the pseudocode pieces related to both schemes.

Figure 4 is inserted here.

The aforementioned music-rules-handling module is used for finding the suitability of the local changes made to each intermediate piece. The employed variable neighbourhood search uses its two neighbourhood schemes in a loop, in the sense that after making a piece local optimized through the first neighbourhood scheme, the result will become local optimized through the second scheme. The loop is terminated when neither of the schemes can make any improvement in the piece.

The rhythm generation module is the second module described in this subsection. This module is much simpler than the pitch generation module. Rhythm, as regular recurring motion, indicates the regulated succession of notes based on their durations.

By arranging pitches and assigning time to each pitch, the SYMPHONY constructs its generated pieces, and the time of pitches define the rhythm. Although a common notation for the time of pitches is that the first shorter duration for each pitch is half of its previous duration, the time ratio between two consecutive notes can be any real number. In effect, the notations of whole note, half note, quarter notes, etc., are only for simplification purposes. The SYMPHONY, however, sticks to these typical notations and generates rhythms randomly based a distribution pattern received as input. Figure 5 shows the pseudocode related to its rhythm generation module.

Figure 5 is inserted here.

This subsection is concluded by describing the chord generation module. Since chords accompany the main notes, in the SYMPHONY, the chord generation module has been intertwined with the module constructing pitches. The concept of chord progression in its simplest possible fashion has been used as the basis of chord generation. That is why the chord generation module of the SYMPHONY simply works based on tonic, subdominant, dominant, and sixth tonal degree of the diatonic scale, preventing dominant to subdominant and subdominant to sixth tonal degree, allowing all other 14 possible progressions, 4*4-2.

Note that based on chord progression rules, among 16 possible progression cases among I (i), IV (iv), V(v), and vi (VI) only two are not allowed, namely V(v)→ IV(iv) & IV(iv)→vi(VI), which have been prevented in the program. Depending on the selected major(minor) scale, I(i), IV(iv), V(v), and vi(VI) chords have randomly been

added to the notes if the corresponding note is part of that chord: with chord I(i) for I(i), V(v) for ii (ii°), I (i) for iii (III), IV(iv) for IV(iv), V(v) for V(v), VI(vi) for vi(VI), and V(v) for vii° (VII), noticing that for harmonic minor scale, V should be used instead of v.

The consonant intervals include octave (2/1), fifth (3/2), fourth (4/3), and major tone (9/8). The significance of major, (5/4), and minor, (6/5), third, considered as imperfect consonances, has been noticed in as early as the thirteenth century. It is worth noting that consonant intervals plus diatonic semitone, 13/12 are all super-particular ratio, i.e. (n+1)/n=1+1/n. In music theory any interval which is not consonant is called dissonance. Based on chord progression and consonant intervals, the chord generation module adds randomly between zero and three chords to each main note so that no two simultaneously played notes are dissonance.

3.3 The timber, dynamics, and other modification modules

As the chord generation module has a strong connection with the pitch generation module, the timber and dynamics modification modules have a robust linkage with the rhythm generation module. The reason is that in the SYMPHONY, a rhythm is a structure and this structure includes not only duration but dynamics and timber as well. Referring to the volume of a note, and the device by which the note is played, dynamics and timber, respectively, make music versatile and more interesting.

With respect to timber, for each of the four voices which can be played simultaneously, the module allows the user to dynamically select one of the 127 various devices, which vary from acoustic grand piano through guitar harmonics to trumpet and flute. On the other hand, dynamics are handled by making a pattern of strong and weak pulses and applying the pattern to the rhythms generated. In such a pattern, each 1 is followed by

several 0’s, with 1’s indicating strong and 0’s indicating weak pulses.

For instance, the pattern (1,0,0,0,0,1,0,0,00) is interpreted as a strong pulse followed by 4 weak pulses, followed by another strong pulse, and then terminated with four weak pulses. Interruption of the rhythmic flow through placing different rhythmic stresses in the places they wouldn't normally occur is not prevented but irregularly used to incorporate verity to the composed pieces. In music theory, the term syncopation refers to such displacement, which has been widely used in many musical styles.

The modification modules can also perform transposition. In music, transposition is about moving a group of notes up or down by any given interval. As an example, if we transpose a G Major chord (G, B, and D) up by a major second, we then have an A Major chord (the notes A, C-sharp, and E). In this way, the corresponding modification module allows the user to dynamically change the key of the major scale in which a piece has been composed.

Also, since music benefits from modified repetition, the SYMPHONY is equipped with a modification module performing improvisation. By modifying and repeating the motif constructed, this module prolongs the duration of the composed piece. The main consideration in the construction of this module is that the more a motif is sophisticated, the more it can be repeated. After all, a motif which is supposed to be repeated in a modified manner several times should have the necessary complexity for holding the listeners’ attentions for all of its successive modified repeats.

When the user selects number of notes in the piece as a small value, say less than 16 notes, this sophistication can hardly be present in the motif. In these cases, the change of tempo is used as a compensation. In other words, the modification module performing improvisation not only can alter pitches but can change the duration of notes

as well.

In effect, not only as an artificial composer, does the SYMPHONY aim at generating automatic music but, as an artificial improviser, it can generate variation of the same inputted or generated music as well. This is performed through a facilitating module used in a graphic user interface for improvising purposes. Figure 6 shows the pseudocode of the improvising module.

Figure 6 is inserted here.

Combining the variable neighbourhood search with evolutionary computation in generating major, minor, and harmonic minor scales can be considered as the main novel point of the SYMPHONY when compared with similar approaches. The SYMPHONY also has a distinguishing feature of combining five capabilities of using input rhythm distribution system, music theory, harmony, symmetry, and multithreading. The integration of these capabilities separates the SYMPHONY from other related systems, as none of those systems has combined all those features.

The SYMPHONY can benefit from the application of symmetry. In applying symmetry in music, based on the definition presented in (Hart, 2009), a combination of repetition (horizontal transformation), transposition (vertical plus horizontal transformation), retrograding (horizontal mirroring), inversion (vertical mirroring), and rotation (vertical plus horizontal mirroring or in simple term “retrograde inversion”) can be used to modify the themes.

Figure 7 is inserted here.

Figure 7 shows examples of symmetry. A point with applying symmetric operations is that their combination is also a symmetric compound operator. For instance, both

rotation and transposition can be applied to adjust a theme. The issue is not simply that the SYMPHONY uses symmetry for adding recognisable patterns to its composed pieces; the deeper issue is that the regularity of such patterns can create expectations for the listener, which easily and mathematically are responded to.

4. Results

This section describes the examples of pieces generated by the SYMPHONY, and novel points regarding their generation. The examples are in two categories, namely (i) simple melodies, and (ii) their corresponding chords. In each category, there are three subcategories related to (i) major, (ii) minor, and (iii) harmonic minor scales, and in each subcategories, there are six pieces composed by the SYMPHONY. Hierarchically, all of these 36, 2*3*6, voice files have been placed in a folder based on their subcategories, and after being zipped, the zipped folder has been uploaded as a resource file for this paper.

In general, the SYMPHONY explores variations to throw a building block of notes, as a theme, towards the listener from multiple perspectives. In this regard, upon composing such a phrase of notes with whatever size requested by the user, the SYMPHONY improvises it, and then plays the original phrase along with its improvised phrases. Three improvised phrases are constructed based on the original phrase, and then these phrases along with the original phrase are played two times to impose symmetry.

The improvisation made is a follows. Randomly, the durations of two neighbouring notes are swapped and the same is true with two successive distances which follow the following condition. The condition is that their previous distance is of the same sign

as those two distances and the two distances do not have a difference more than two semitones. Towards enforcing symmetry, each copy can be played in lower or higher octave.

The underlying foundation of creating these examples is the employing of music rules, which are mainly due to Fax's work which has been translated in English and edited by Alfred Mann, as a Professor of Musicology, in 1965 (Mann, 1965). Rules in this work have been stated as a form of dialogue between a master and a pupil.

In creating these examples, the SYMPHONY, as a stochastic formal system working based on a genetic algorithm, uses an alphabet of music, $\sum$ with the size of α, representing notes in $\alpha/7$ consecutive octaves. To be musically consistent, the restriction is that α is divisible by 7, the number of notes in an octave. Constructing stochastic formulas, the SYMPHONY is guided by music theory and uses this theory as the base of its rule manipulating mechanism in creating a music composition (formulas).

By the term of "formal system", here we mean a well-defined system of abstract thought composed of a set of symbols as well as a collection of rules to manipulate these symbols in integrating them into meaningful formulas (compositions). In all of the examples provided, the combination of genetic algorithm and the variable neighbourhood search has been able to minimise the penalties in the level of o or 1. It is worth noting that when the penalty (anti-score) for a generated piece is 1, in the generation of that piece only one rule has been violated.

It seems that violating an insignificant number of rules does not degrade the quality of pieces. The reason can be the fact that creativity cannot be captured by pure rules, and when some rules are frequently broken, still attaining an acceptable level of musicality

is possible. After all, the key issue with any artificial art, in general, and music composition, in particular, is to impose some degree of unpredictability, which can be attained through inserting insignificant chaos into significant order.

Despite the fact that the SYMPHONY can potentially create music pieces with any size but for avoiding any complexity and allowing listeners to track recognizable patterns, in the examples discussed, the restriction of 16 has been placed on the number of notes generated. Based on this restriction, the procedure has created an original phrase with 16 notes and 3 improvised phrases of it. Since all these notes are played twice, each example includes 128, ((3+1)*16)*2, notes. Despite having the same number of notes, since the duration of these notes has been produced in the rhythm generation module based on the given distribution randomly, the pieces have different duration.

In the examples provided the chords surrounding the main notes can be heard separately, and as mentioned, they are in separate folders. For generating chords, music theory has been used for establishing rules to indicate whether or not several voices are harmonic. The provided chord examples have been generated as a result of the fact that after composing the main notes of a piece, the SYMPHONY generates up to three chord notes associated with each main note. For the simplicity purposes, the maximum of four voices generated are not independent of one another as it is the case with polyphonic music. On the contrary, the chord notes depend on the main notes.

For measuring the results of different components on the performance of the procedure, factors, one by one, have been deactivated and the behaviour of the system has been tested based on the appealing status of the pieces generated. It is worth noting that at each instance only one factor has been deactivated. Moreover, because of the complexity involved with such measuring, no continuous measure has been used and

only a binary number has been employed, indicating appealing or not.

The factors tested are: (i) the musically informed crossover, (ii) the musically informed mutation, (iii) the indiscriminate crossover, (iv)the indiscriminate mutation, (v) the rhythm generation distribution, (vi) music theory as the base for constructing fitness value, and (vii) the chord generating module. The behavior of the system is as follows.

With removing the factor (v) or (vi) the pieces generated were no more appealing. Indeed, the integration of these two factors is a requirement for the SYMPHONY to produce appealing pieces. The other five factors were not as crucial as those two factors, as with their removal and keeping all the remaining six factors intact, the generated pieces were still appealing.

As mentioned, the crucial factors distinguished were the rhythm distribution and music theory, which was used as the base for constructing fitness value. Indeed, the experiments shows that rhythm plays a key role in generating artificial music and well-constructed melodic pieces without appropriate rhythms cannot perform well. One of the reasons is that appropriate rhythmic structures change static pieces to dynamic pieces.

The fact that using music theory as the base for constructing fitness value has a crucial role in generating appealing pieces is not surprising. After all, Fuxian rules, as the base of music theory, play a key role in generating high quality music. The establishment of Fuxian rules more than anything else suggests that there are universal aesthetic principles governing the development of high quality music. By using these rules in its fitness function, the SYMPHONY evolves dull pieces into lively pieces through making suitable intervallic changes.

As two useful features, the SYMPHONY (i) provides an opportunity for the user to see the notes while they are played, and (ii) enables the users to input a piece through playing it on a computer keyboard. Figures 8 and 9 show how the computer keyboard can be used by the user for playing notes and how saving capability can combine the played notes with the generated ones. Figure 9 also depicts how the SYMPHONY allows the user to select the parameters through a simple text file while the program is being executed. Controllable parameters not seen in Figure 9 have been represented in Figure 10.

Figures 8, 9, and 10 are inserted here.

The improvising module can improvise any given piece from the pieces that have been played through the computer keyboard through the generated pieces to the hybrid pieces. The reason why the examples include only those constructed by the program and not hybrids is twofold. First, playing on the computer keyboard and combining the played piece with constructed piece, as a hybrid, has a quality dependent on the piece played on the keyboard. Second, such hybrids are based on the users' musical preference.

5. Conclusion

With art, in general, and music, in particular, the issue is not simply that algorithmic thinking provides the development of innovative and reputable techniques for artistic tasks. The deeper issue is that such thinking presents a medium to capture artistic notions in the form of computer programs which can be enhanced, significantly extending the range of artistic capabilities.

The same within other forms of art which rely on the combination of consent and

surprize, in music the lack of conforming to musical rules creates unpleasant pieces and at the same time anything predictable and mechanical causes the listeners to lose their attention.

The evolutionary search technique makes a delicate trade-off between consent and surprise. This trade-off is achieved through the fact that one of the neighbourhood schemes used in its variable neighbourhood scheme is musically-informed whereas the other is not.

The combination of these two opposite neighbourhood schemes enables the SYMPHONY to search the solution space effectively. Merging intervallic and rhythmic structures towards fulfilling the expectations of listeners, the SYMPHONY manages both dynamics and timber and plays its composed pieces in a multi-threading environment which can manage up to four voices. The capability of assigning different music devices to different voices and altering the devices while the piece is being played adds to the versatility of the procedure. With respect to improving the performance of SYMPHONY, two main directions are proposed.

First, the employed chord generation mechanism can be replaced with generating polyphonic voices. In polyphonic music, a piece consists of two or more melodies, also called voices, which are played simultaneously. Whereas these melodies should be independent in terms of having different counters and possibly rhythms, they all need to be harmonically interdependent. In this case, the genetic algorithm employed in arranging pitches, albeit with a different fitness function, needs to be used in adjusting these simultaneous voices.

Second, a neural network can be developed for evaluating the fitness of pieces. This new evaluator can then be employed along with current music rules evaluator, with

each of them having different weights calculated experimentally. Since experimental results showed the importance of fitness function as one of the two crucial factors employed in the SYMPHONY, the integration of such a neural network with the current fineness function can be of paramount importance.

References

A-Rifaie, M. M., Fol Leymarie, F., Latham, W., & Bishop, M. (2017). Swarmic autopoiesis and computational creativity. *Connection Science*, pp. 1-19.

Alarcón, D. (2013). *Rule-based Music Composition using Cantus Firmus*. The Australian National University

Aloupis, G., Fevens, T., Langerman, S., Matsui, T., Mesa, A., Nuñez, Y., . . . Toussaint, G. (2006). Algorithms for computing geometric measures of melodic similarity. *Computer Music Journal, 30*(3), pp. 67-76.

Andreatta, M. (2011). Constructing and formalizing tiling rhythmic canons: A historical survey of a "mathemusical" problem. *Perspectives of New Music, 49*(2), pp. 33-64.

Bel, B. (1998). Migrating musical concepts: an overview of the bol processor. *Computer Music Journal*, pp. 56-64.

Boden, M. A. (2016). Skills and the appreciation of computer art. *Connection Science, 28*(2), pp. 131-138.

Chemillier, M. (2004). Periodic musical sequences and Lyndon words. *Soft Computing, 8*(9), pp. 611-616.

Chen, Y.-p. (2007). Interactive music composition with evolutionary computation. *NCLab Tech. Report*

Clifford, R., & Iliopoulos, C. (2004). Approximate string matching for music analysis. *Soft Computing, 8*(9), pp. 597-603.

Conklin, D. (2002). Representation and discovery of vertical patterns in music *Music and Artificial Intelligence* (pp. 32-42): Springer.

Conklin, D., & Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research, 24*(1), pp. 51-73.

Dahlstedt, P., & McBurney, P. (2006). Musical agents: toward computer-aided music composition using autonomous software agents. *Leonardo, 39*(5), pp. 469-470.

de Vega, F. F., Cruz, C., Navarro, L., Hernández, P., Gallego, T., & Espada, L. (2014). Unplugging Evolutionary Algorithms: an experiment on human-algorithmic creativity. *Genetic Programming and Evolvable Machines, 15*(4), pp. 379-402.

Demaine, E. D., Gomez-Martin, F., Meijer, H., Rappaport, D., Taslakian, P., Toussaint, G. T., . . . Wood, D. R. (2009). The distance geometry of music. *Computational geometry, 42*(5), pp. 429-454.

Diaz-Jerez, G. (2011). Composing with Melomics: Delving into the computational world for musical inspiration. *Leonardo Music Journal, 21*, pp. 13-14.

Eck, D. (2001). A positive-evidence model for rhythmical beat induction. *Journal of New Music Research, 30*(2), pp. 187-200.

Gale, E., Matthews, O., Costello, B. d. L., & Adamatzky, A. (2013). Beyond Markov Chains, Towards Adaptive Memristor Network-based Music Generation. *arXiv preprint arXiv:1302.0785*

Gwee, N. (2013). Music, Mathematics, and Microcomputers. *GSTF Journal on Computing, 3*(3)

Halkiopoulos, C., & Boutsinas, B. (2012). Automatic Interactive Music Improvisation based on Data Mining. *International Journal on Artificial Intelligence Tools, 21*(04), p 1250016.

Hart, V. (2009). *Symmetry and transformations in the musical plane.* Proceedings of Bridges 2009: Mathematics, Music, Art, Architecture, Culture.

Herremans, D., & Sörensen, K. (2012). Composing first species counterpoint with a variable neighbourhood search algorithm. *Journal of Mathematics and the Arts, 6*(4), pp. 169-189.

Hirata, K., & Aoyagi, T. (2003). Computational music representation based on the generative theory of tonal music and the deductive object-oriented database. *Computer Music Journal, 27*(3), pp. 73-89.

Horner, A., & Goldberg, D. E. (1991) *Genetic algorithms and computer-assisted music composition*. Paper presented at the International Conference on Genetic Algorithms

Hsu, J.-L., Liu, C.-C., & Chen, A. L. (2001). Discovering nontrivial repeating patterns in music data. *Multimedia, IEEE Transactions on, 3*(3), pp. 311-325.

Jo, J.-y., Kim, Y.-h., Kang, H.-j., & Lee, J.-s. (2007). *Chord-based musical composition and incorporating it into auto-accompaniment instrument.* Future Generation Communication and Networking (FGCN 2007).

Karydis, I., Nanopoulos, A., & Manolopoulos, Y. (2007). Finding maximum-length repeating patterns in music databases. *Multimedia Tools and Applications, 32*(1), pp. 49-71.

Liu, C.-H., & Ting, C.-K. (2012). *Polyphonic accompaniment using genetic algorithm with music theory*. Evolutionary Computation (CEC), 2012 IEEE Congress on.

Liu, X. F., Chi, K. T., & Small, M. (2010). Complex network structure of musical compositions: Algorithmic generation of appealing music. *Physica A: Statistical Mechanics and its Applications, 389*(1), pp. 126-132.

Loaiza, J. M. (2016). Musicking, embodiment and participatory enaction of music: outline and key points. *Connection Science, 28*(4), pp. 410-422.

Mann, A. (1965). The study of counterpoint. *WW Norton, New York, 1943*, p 1971.

Mazzola, G. (2012). Thinking music with precision, depth, and passion. *Journal of Mathematics and Music, 6*(2), pp. 83-91.

Moroni, A., Manzolli, J., Von Zuben, F., & Gudwin, R. (2000). Vox populi: An interactive evolutionary system for algorithmic music composition. *Leonardo Music Journal, 10*, pp. 49-54.

Munoz, E., Cadenas, J., Ong, Y. S., & Acampora, G. (2012). Memetic Music Composition. *IEEE Transactions on Evolutionary Computation, 11*(4)

Papadopoulos, A. (2014). Mathematics and group theory in music. In A. Papadopoulos & S.-T. Yau (Eds.), *Handbook of Group actions* (Vol. II): Higher Education.

Read, R. C. (1997). Combinatorial problems in the theory of music. *Discrete Mathematics, 167*, pp. 543-551.

Rolland, P.-Y. (1999). Discovering patterns in musical sequences. *Journal of New Music Research, 28*(4), pp. 334-350.

Shan, M.-K., & Chiu, S.-C. (2010). Algorithmic compositions based on discovered musical patterns. *Multimedia Tools and Applications, 46*(1), pp. 1-23.

Smaill, A., Wiggins, G., & Harris, M. (1993). Hierarchical music representation for composition and analysis. *Computers and the Humanities, 27*(1), pp. 7-17.

Tardón, L. J., Barbancho, I., Barbancho, A. M., & Roig, C. (2014). A probability model for key analysis in music. *Knowledge-Based Systems, 67*, pp. 169-179.

Todd, P., & Werner, G. (1999). Frankensteinian methods for evolutionary music composition In N. Griffith & P. Todd (Eds.), *Musical networks: parallel distributed perception and performance*: Cambridge, Mass.: MIT Press. .

Toussaint, G. (2010). Computational geometric aspects of rhythm, melody, and voice-leading. *Computational geometry: Theory and applications, 43*(1), pp. 2-22.

Toussaint, G. T. (2003). Algorithmic, geometric, and combinatorial problems in computational music theory. *Proceedings of X Encuentros de Geometria Computacional*, pp. 101-107.

Vargas, F. V., Fuster, J. A., & Castanón, C. B. (2014). *Artificial musical pattern generation with genetic algorithms*. http://www.researchgate.net/publication/268447156

Volchenkov, D., & Dawin, J. R. (2012). *Musical Markov Chains.* International Journal of Modern Physics: Conference Series.

Witten, I. H., Manzara, L. C., & Conklin, D. (1994). Comparing human and computational models of music prediction. *Computer Music Journal, 18*(1), pp. 70-80.

****

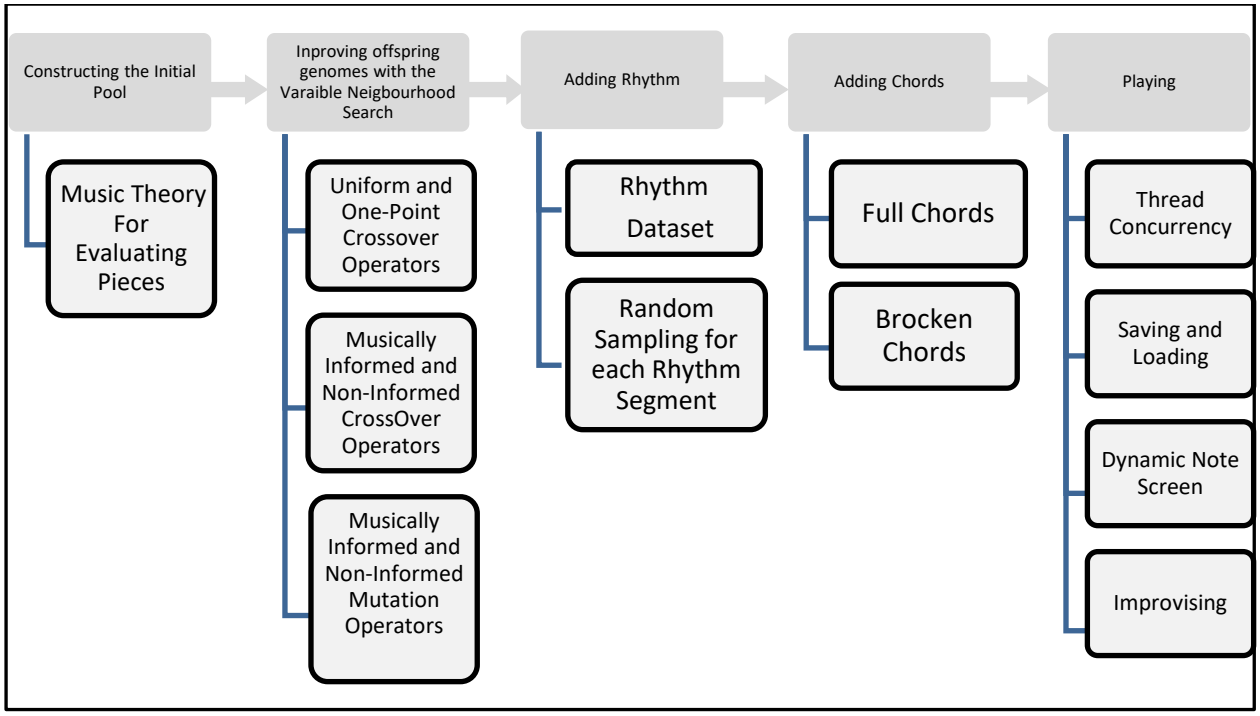

Figure 1. The schematic representation of the SYMPHONY and the machinery it employs for generating and playing music

```
PROCEDURE SYMPHONY() //Composing based on the genetic algorithm
   Read the input Parameters.
   Call GA()  to improve the pool of pitch sequences based on music theory.
   Select the best pitch sequences in the pools generated by GA(), and call it PS.
   Based the input parameter, randomly select a number of rhythms from the rhythm dataset.
   Combine PS with the selected rhythms and call it the First Phrase.
   Add chords to the First Phrase.
   Improvise the first phrase and create three different phrases based on it.
   Two times play the phrases with their chords and show the main notes on the screen while playing.
ENDPROCEDURE



PROCEDURE GA() //the genetic algorithm employed
  Create a random pool of pitch sequences and put them in vector P.
  FOR k = 0 TO NumberOfGeneration
        Create a new empty vector Q.
        FOR i = 0 TO PoolSize-1
            Shuffle the vector P.
            NumberOfGeneratedChildren=0
            SET J, as an index for the vector P, to 0.
            WHILE NumberOfGeneratedChildren  is not equal to PoolSize.
                SET Parent1 to P[j % PoolSize] and Parent2 to P[(j+1) %  PoolSize].
                Do Crossover on Parent1 and Parent2 and call the result Offspring.
                Perform mutation on Offspring.
                Compute the fitness of Offspring based on music theory.
                Put Offspring in vector Q.
                Add one to NumberOfGeneratedChildren
                Add one to J.
            ENDWHILE
           SET vector S to the combination of vectors P and Q.
           Sort vector S in descending manner based on the fitness of its pitch sequences.
           Select as many as PoolSize top pitch sequences of S and put them in P.
        ENDFOR
  ENDFOR
ENDPROCEDURE
```

Figure 2. The pseudocode of the SYMPHONY and its genetic algorithm component

```
PROCEDURE CreatePitchArrangement() //Creating a pitch arrangement musically informed
    SET Sign to either 1 or -1 randomly.
    SET i to 0. //I shows the number of notes generated.
    WHILE   (i < n) //n is an input parameter and shows the number of notes to be generated
        SET Sign to ( -1*Sign)  //based on the []() scheme the directions alternate .
            SET U to 0. // U is used for biased random generation of distances
            Generate a random number between 1 and 100 inclusively and call it R.
            SET BlockSize to 0.
            WHILE  (R > U) // For generating BlockSize with the given chance in  vector Pcntg.
                Add BlockSize by 1.
                Add U by Pcntg[BlockSize] //note that   Pcntg[1]+...Pcntg[5]=100.
            ENDWHILE
            FOR k=1 to  BlockSize //For generating a homogenous block with the size of BlockSize.
                IF (i <n)
                    SET Dist[i] to Sign.
                    Add 1 to i.
                ENDIF
            ENDFOR
            Generate a random number between 1 and 100 inclusively and call it R.
            IF (  R <  OppositePcntg  & i < n ) // Adding a possible non-step in reverse direction
                    SET Sign to ( -1*Sign)
                    Generate a random number between 2 and 5 inclusively and call it Q.
                    SET Dist[i] to (Sign*Q).
                    Add 1 to i.
            ENDIF
    ENDWHILE
    Convert distance vector Dist to note vector V.
    Return V
ENDPROCEDURE
```

Figure 3. The pseudocode of the musically informed pitch arrangement module

```
PROCEDURE BasicLocalSerach(U) //vector U is the initial starting point.
      Convert U to a vector of distances, D, based the distance of consecutive notes in U.
      Compute the quality of D based on music theory.
      FOR i=1 to size of D.
           SET vector D' to D.
           Change D'[i] in the  given range randomly.
           Compute the quality of D' based on music theory.
           IF the quality of D' is more than the quality of D then SET D to D'.
      ENDFOR
      Convert the vector of disnatures, D, to V as the vector of notes.
      RETURN V
ENDPROCEDURE


PROCEDURE  MusicallyInformedLocalSearch(U) //vector U is the initial starting point.
       Convert U to a vector of distances, D, based the distance of consecutive notes in U.
       Compute the quality of D based on music theory.
       FOR i=1 to size of D.
           SET vector D' to D.
           IF (D'[i-1]*D'[i]  > 0  &  ABS(D'[i-1]) > 1 & ABS(D'[i]) > 2)
                IF D'[i] > 0
                      Set D'[i] to -1.
                ELSE
                     Set D'[i] to  1.
                ENDIF
           ENDIF
           Compute the quality of D' based on music theory.
           IF the quality of D' is more than the quality of D then SET D to D'.
       ENDFOR
      Convert the vector of disnatures, D, to V as the vector of notes.
      RETURN V
ENDPROCEDURE
```

Figure 4. The pseudocode of the basic and musically informed neigbourhood schemes employed in the variable neigbourhood search

```
PROCEDURE RG() //Rhythm generator
      FOR i= 1 to k, //k, as an input parameter, shows the number of measures with different rhythms.
          SET Sum to 0 //Sum shows the duration of a measure with the number of sixteenths
          SET j to 1 // j is an index for the rhythm vector
          WHILE (Sum != 16)
                Generate a random number between 1 and 100 inclusively and call it R.
                SET U to 0.
                 SET Dur to 0.
                 WHILE  (R > U)
                        Add Dur by 1;
                        Add U by Percentage[Dur] //note that   Percentage[1]+..... Percentage[8]=100
                       //Dur 1 shows eights, 2 shows quarters, 4 shows halves, and 8 shows wholes.
                 ENDWHILE
                IF (Sum + Dur is not greater than 16)
                      SET Rhythm[i, j] to Dur.
                      Add j by 1.
                      Add Sum by Dur.
                ENDIF
           ENDWHILE
          WHILE  (j  ≤ 16)
                 SET Rhythm[ i , j] to 0.
                 Add j by 1.
          ENDWHILE
     ENDFOR
ENDPROCEDURE
```

Figure 5. The pseudocode of the RG (Rhytm Generator) module

```
PROCEDURE Improvise() //modifying distances consistently and musically informed
    Calculate the difference between every two consecutive notes of the piece to be improvised.
    SET Diff, as a vector, to calculated differences.
    WHILE (True) //as soon as a proper location is found in Diff vector, this while loop will be broken.
      SET Location randomly to a number between 1 and the size of Diff vector.
      SET FirstDiff to Diff [Location -1]
      SET SecondDiff  to Diff [Location]
            IF (FirstDiff = 0 &  SecondDiff = 0 )
                  SET CurrentCase = 1.
                  BREAK WHILE
             ENDIF
            IF (ABS (FirstDiff) < 3 & ABS (SecondDiff) < 3)
                 IF (FirstDiff * SecondDiff  < 0)  SET  CurrentCase = 2
                 IF (FirstDiff * SecondDiff  > 0)  SET CurrentCase = 3
                 IF (FirstDiff * SecondDiff  != 0) BREAK WHILE
             ENDIF
             IF (FirstDiff * SecondDiff  < 0 & ABS (FirstDiff + SecondDiff ) <5
                          & (ABS (FirstDiff ) > 3 | ABS (SecondDiff) > 3))
                  SET CurrentCase = 4.
                  BREAK WHILE
             ENDIF
    ENDWHILE
    SWITCH (CurrentCase)
          CASE 1:
               SET  FirstElement to 2 and  SecondElement to  -2.
               With the chance of 50 percent  change the signs of both
                    FirstElement and SecondElement.
          CASE 2:
               SET  FirstElement to SecondDiff.
               SET  SecondElement to FirstDiff.
          CASE 3:
               SET  FirstElement to FirstDiff + SecondDiff.
               IF  (FirstElement > 0)
                    ADD FirstElement BY 2.
                    SET SecondElement  TO -2
               ELSE
                    ADD FirstElement  BY -2
                    SET SecondElement TO 2
               ENDIF
           CASE 4:
                    SET FirstElemen TO FlOOR ((FirstDiff + SecondDiff)/2).
                    SET SecondElement  TO FirstDiff + SecondDiff – FirstElement.
      ENDSWITCH
      FOR i=2  TO  size of Diff
          IF (Diff[i-1]= FirstDiff & Diff[i]= SecondDiff )
              SET Diff[i-1] TO FirstElement
              SET Diff[i]   TO  SecondElement
          ENDIF
      FOREND
      Convert vector Diff to a note vector showing improvised notes.
ENDPROCEDURE
```

Figure 6. The pseudocode of the improvising module

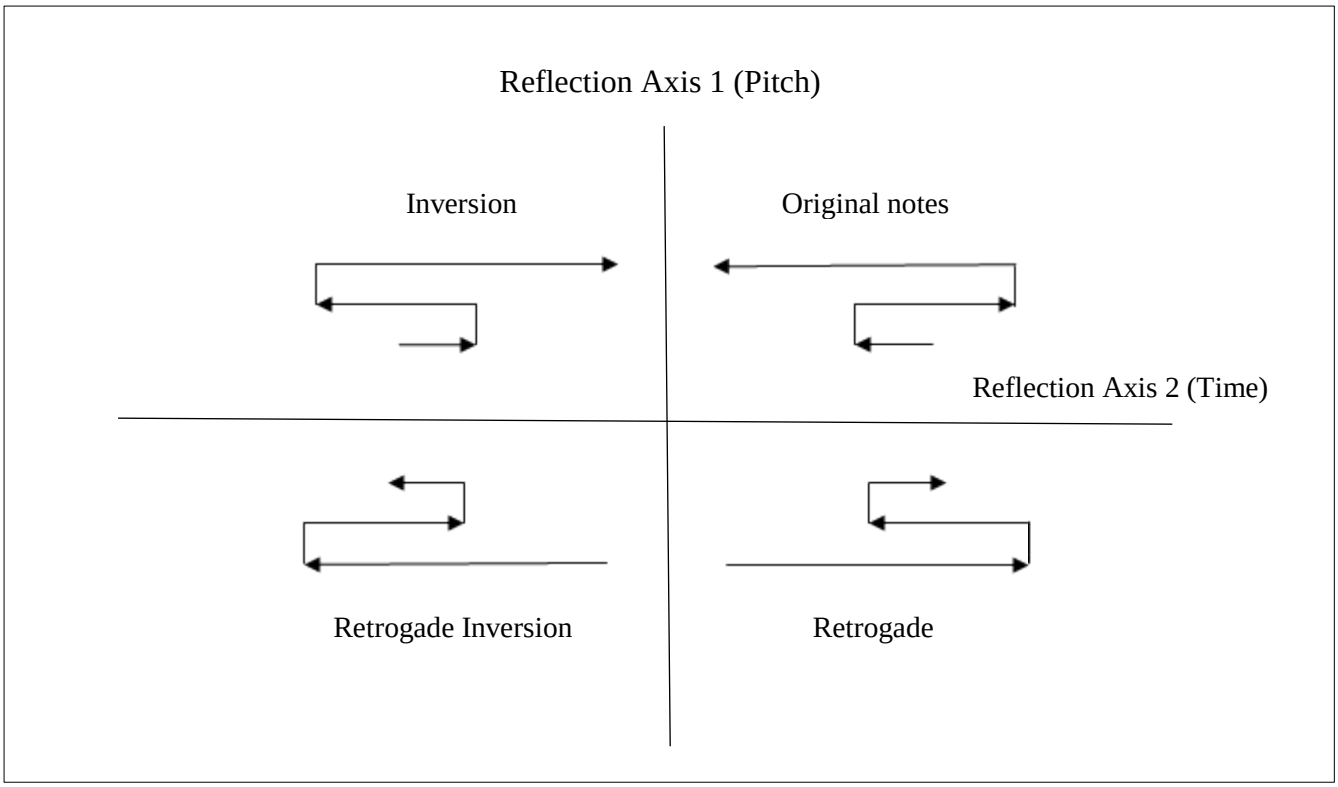

Figure 7. Retrogade-Inversion as 180° rotation of the original notes (combination of vertical and horizontal mirroring)

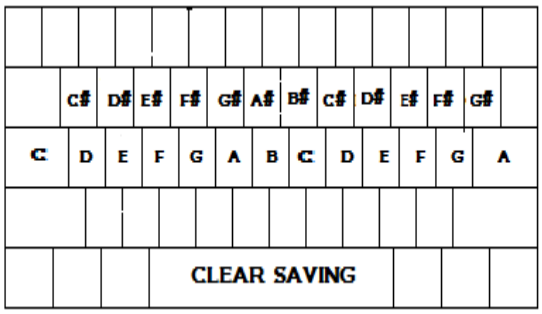

Figure 8. Using the computer keyboard by the SYMPHONY for playing notes and combining them with the composed music

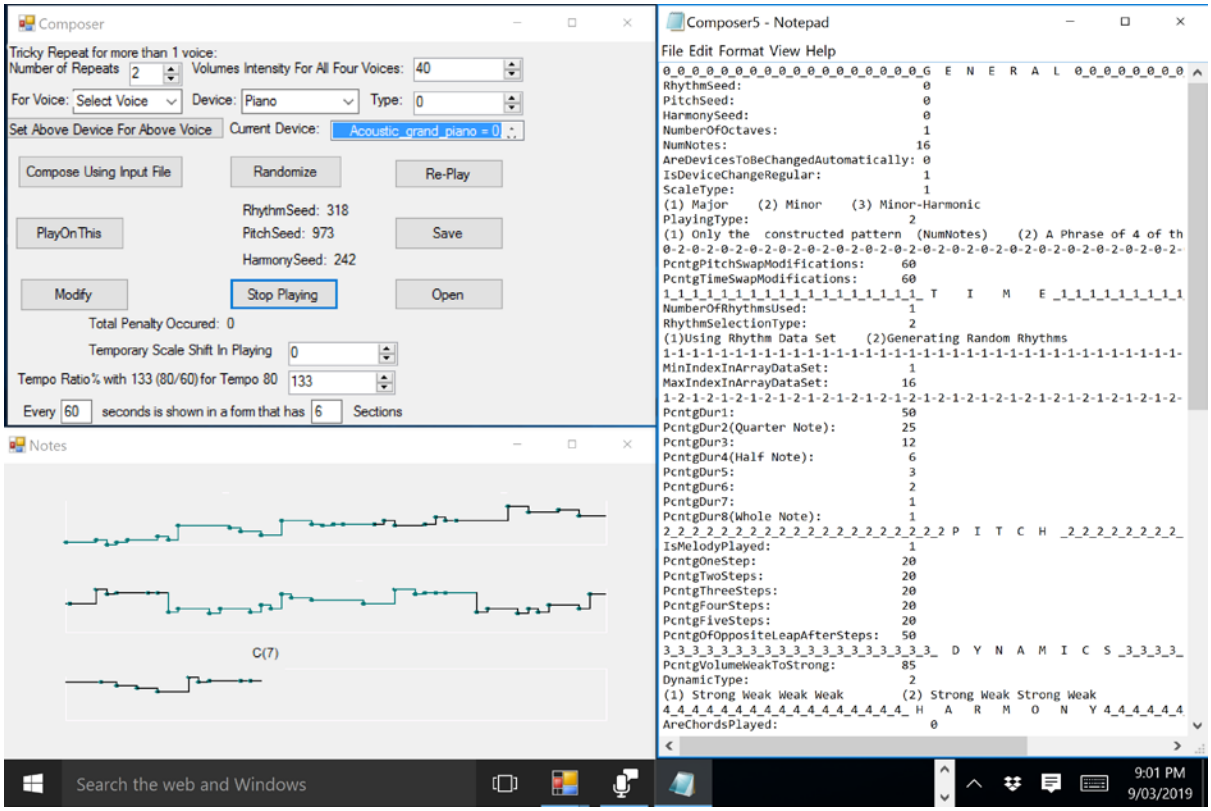

Figure 9. The screen seen and the parameters controllable by the user while SYMPHONY composes and plays.

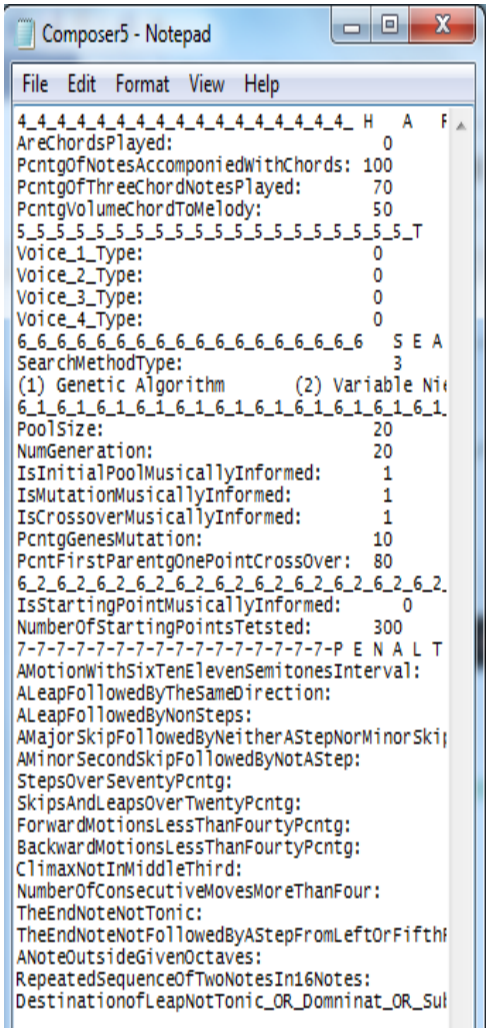

Figure 10. The rest of the controllable parameters (not seen in Figure 9)

Table 1. The definition of terms needed to understand the objective function of the variable neighborhood search

Music Term	Definition
Scale	Any set of musical notes ordered by frequency or pitch
Diatonic	A scale with seven pitch classes
Scale Degree	A number given to a particular note of a scale to designate its location relative to the first note of the scale.
Octave	An interval by which any two notes separated have the same scale degree
Semitone	An interval with the size of a twelfth of an octave
Pitch	A note with the ratio between wave frequency of two consecutive notes being equal to $\sqrt[12]{2}$
Tonic	The first scale degree of a diatonic scale
Leading-Tone	The seventh scale degree of a diatonic scale
Step	The interval between two consecutive scale degrees
Skip	The interval of two steps, which can be of three (minor) or four semitones (major)
Leap	Any interval larger than skip
Consonance	Pleasant intervals like 0, 3, 4, 5, 7, 8, 9, and 12 semitones
Dissonance	Unpleasant intervals like 6, 10, and 11 semitones
Climax	Part of a composition where music reaches its highest pitch

Table 2. The possible discouraged events and their penalties in calculating the objective function of the variable neighbourhood search

Event	Penalty
The total number of steps is over seventy five percent of all the notes.	1 Per Over
The total number of skips and leaps is over twenty percent of all the notes.	1 Per Over
A leap is followed by a note in the same direction.	1 Per Occurrence
A leap is not followed by a step.	1 Per Occurrence
A major-skip is followed by neither a step nor a minor-skip.	1 Per Occurrence
A minor-skip is not followed by a step.	1 Per Occurrence
A note is outside given octaves.	1 Per Occurrence
A sequence of two notes is repeated in the range of 16 notes.	1 Per Occurrence
A movement in same direction with six, ten, or eleven semitones interval occurs.	1 Per Occurrence
The number of consecutive notes in the same direction is more than four.	1 Per Occurrence
A note outside the given octaves.	1 Per Occurrence
The destination of a leap is not of 1, 4, 5, or 6 tonal degrees.	1 Per Occurrence
Forward steps are under forty percent.	1 Per Under
Backward steps are under forty percent.	1 Per Under
A climax is outside the middle third	1
The end note is non-tonic	1
The end note is not preceded by a step from left or fifth from right.	1